DETAILED ACTION

1. This action is in response to Applicant's amendments filed December 22, 2009.

2. Claims 1, 7, and 13 have been amended.

3. Claims 1, 2, 4, 5, 7, 8, 10, 11, 13, 14, 16, 17, and 19-24 are allowed.

Examiner's Amendment

4. An examiner's amendment to the record appears below. Should the changes and/or additions be unacceptable to applicant, an amendment may be filed as provided by 37 CFR 1.312. To ensure consideration of such an amendment, it MUST be submitted no later than the payment of the issue fee.

Authorization for this examiner's amendment was given in a telephone interview with Kendrick Lo on March 5, 2010.

The application has been amended as follows:

IN THE CLAIMS

1. (currently amended) A device comprising:

a memory unit including executable software;

a plurality of class files stored in the memory unit; and,

a computing unit connected to the memory unit, the computing unit being able to execute a Java Virtual Machine, the computing unit configured to ~~execute software for generating~~ generate two or more files from the plurality of class files by combining elements from the plurality of class files without duplication of entries for reducing storage space[[,]];

wherein the generated files are directly interpretable by the Java Virtual Machine; [[, and]]

wherein the number of the generated files is less than the number of the plurality of class files; [[, and]]

wherein a given generated file comprises:

a constant pool created by combining constant pool entries from two or more of the plurality of class files without duplication of entries;

a byte codes and information structure created by combining byte codes and information structure entries from the two or more of the plurality of class files; and,

a fixup table for providing information to the Java Virtual Machine, wherein the Java Virtual Machine uses the information in the fixup table to resolve ~~for resolving~~ at least one entry in the given generated file at link time; [[, and]]

wherein at least two of the generated files are generated as sibling files in a common sibling group; [[,]]

wherein each of the sibling files comprises a sibling list for listing other sibling files in the common sibling group; [[,]]

wherein cross-references between the sibling files in the common sibling group are indicated using hard offsets; [[,]] and

wherein references to files that are not part of the common sibling group are indicated using symbolic references.

7. (currently amended) A method [[for]] of generating sibling files, the method comprising:

generating two or more files from a plurality of class files having constant pools on a computing unit by combining elements from the plurality of class files without duplication of entries for reducing storage space, wherein the generated files are directly interpretable by a Java Virtual Machine, and wherein the number of the generated files is less than the number of the plurality of class files; [[,]]

wherein said generating two or more files comprises, [[and]] for a given generated file ~~the method comprises~~:

identifying class files with common entries in the constant pools of the class files;

generating a constant pool for the given generated file by combining constant pool entries from the plurality of class files with common entries without duplication;

generating the byte codes and information structure for the given generated file by combining byte codes and information structure entries from the plurality of class files; and,

generating a fixup table for providing information to [[a]] the Java Virtual Machine, wherein the Java Virtual Machines uses the information in the fixup table to resolve ~~for resolving~~ at least one entry in the given generated file at link time; [[and]]

wherein ~~the method further comprises generating~~ at least two of the generated files are generated as sibling files in a common sibling group by providing a sibling list for each of the sibling files listing other sibling files in the common sibling group; [[,]]

wherein cross-references between the sibling files in the common sibling group are indicated using hard offsets; [[,]] and

wherein references to files that are not part of the common sibling group are indicated using symbolic references.

13. (currently amended) A storage medium storing computer executable instructions that when executed by a computing unit, configure the computing unit to:

generate[[s]] one or more files from a plurality of class files by combining elements from the plurality of class files without duplication of entries for reducing storage space, wherein the generated files are directly interpretable by a Java Virtual Machine, and wherein the number of the generated files is less than the number of the plurality of class files;

~~executable software comprises code for generating~~ wherein a given generated file comprises ~~to include~~:

a constant pool created by combining constant pool entries from two or more of the plurality of class files without duplication of entries;

a byte codes and information structure created by combining byte codes and information structure entries from the two or more of the plurality of class files; and,

a fixup table for providing information to [[a]] the Java Virtual Machine, wherein the Java Virtual Machine uses the information in the fixup table to resolve ~~for resolving~~ at least one component of the given generated file at link time;

wherein ~~the executable software further comprises code for generating~~ at least two of the generated files are generated as sibling files in a common sibling group; [[,]]

wherein each of the sibling files comprise a sibling list for listing other sibling files in the common sibling group; [[,]]

wherein cross-references between the sibling files in the common sibling group are indicated using hard offsets; [[,]] and

wherein references to files that are not part of the common sibling group are indicated using symbolic references.

Examiner's Statement of Reason(s) for Allowance

5. The following is an examiner's statement of reasons for allowance:

Baentsch discloses a CAP file format that comprises a text section, a data section, and a fixup table (modified constant pool). Baentsch's CAP format divides the modified constant pool into internal information for linking (this internal information is removed from the modified constant pool after linking) and external information for late code binding. Swetland discloses a unified programming object with a shared constant pool that contains global constant pool entries mapped from local constant pool entries for multiple class files. In simpler terms, Swetland creates a single unified programming object by combining multiple class files without duplicating entries in the global constant pool. Maatta discloses archiving and splitting Java applications using zip or jar file formats. However, none of the references taken either along or in combination teaches the features in such a manner as recited in the independent claims. Specifically the prior art does not teach a "computing unit being able to execute a Java Virtual Machine, the computing unit configured to generate two or more files from the plurality of class files by combining elements from the plurality of class files without duplication of entries for reducing storage space; wherein the generated files are directly interpretable by the Java Virtual Machine; wherein the number of the generated files is less than the number of the plurality of

class files; wherein a given generated file comprises: a constant pool created by combining constant pool entries from two or more of the plurality of class files without duplication of entries; a byte codes and information structure created by combining byte codes and information structure entries from the two or more of the plurality of class files; and, a fixup table for providing information to the Java Virtual Machine, wherein the Java Virtual Machine uses the information in the fixup table to resolve at least one entry in the given generated file at link time; wherein at least two of the generated files are generated as sibling files in a common sibling group; wherein each of the sibling files comprises a sibling list for listing other sibling files in the common sibling group; wherein cross-references between the sibling files in the common sibling group are indicated using hard offsets; and wherein references to files that are not part of the common sibling group are indicated using symbolic references." as recited in claim 1 . The prior art also does not teach "generating two or more files from a plurality of class files having constant pools on a computing unit by combining elements from the plurality of class files without duplication of entries for reducing storage space, wherein the generated files are directly interpretable by a Java Virtual Machine, and wherein the number of the generated files is less than the number of the plurality of class files; wherein said generating two or more files comprises, for a given generated file: identifying class files with common entries in the constant pools of the class files; generating a constant pool for the given generated file by combining constant pool entries from the plurality of class files with common entries without duplication; generating the byte codes and information structure for the given generated file by combining byte codes and information structure entries from the plurality of class files; and, generating a fixup table for providing information to the Java Virtual Machine, wherein the Java Virtual

Machines uses the information in the fixup table to resolve at least one entry in the given generated file at link time; wherein at least two of the generated files are generated as sibling files in a common sibling group by providing a sibling list for each of the sibling files listing other sibling files in the common sibling group; wherein cross-references between the sibling files in the common sibling group are indicated using hard offsets; and wherein references to files that are not part of the common sibling group are indicated using symbolic references." as recited claim 7 and substantially recited in claim 13.

Any comments considered necessary by applicant must be submitted no later than the payment of the issue fee and, to avoid processing delays, should preferably accompany the issue fee. Such submissions should be clearly labeled "Comments on Statement of Reasons for Allowance."

Conclusion

6. The prior art made of record and not relied upon is considered pertinent to applicant's disclosure. Factor discloses packaging data structures into a common data pool into a combined output file. Swetland discloses combining bytecodes into a bundle for the purpose of reducing memory requirements for object-oriented programs.

7. Any inquiry of a general nature or relating to the status of this application or concerning this communication or earlier communications from the examiner should be directed to Kimberly Jordan whose telephone number is 571-270-5481. The examiner can normally be reached on Monday-Friday 9:30am-5pm EST. If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Hyung Sough can be reached on 571-272-6799.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free). If you would like assistance from a USPTO Customer Service Representative or access to the automated information system, call 800-786-9199 (IN USA OR CANADA) or 571-272-1000.

Kimberly Jordan
March 22, 2010
/K.J./
Examiner, Art Unit 2194

/Hyung S. Sough/

Supervisory Patent Examiner, Art Unit 2194
03/11/10